Self-Healing Intrusion Detection System Concept

Viktoriya Degeler, Richard French and Kevin Jones
Airbus Group Innovations
Newport, UK
{*viktoriya.degeler, richard.french, kevin.jones*}*@eads.com*

***Abstract*—One of the most important open research questions in cyber security is the ability of the system to intelligently detect new previously unseen threats, and react to them in such a way that minimizes the damage and potentially removes the threat altogether. This paper presents a concept of an intrusion detection system based on anomaly detection and danger signals recognition. The system monitors events in the environment, constructs patterns of event sequences, and finds strange and anomalous patterns. If any dangerous symptoms are detected in the environment, the system matches them to the timeline of events and finds a pattern that may have caused the symptoms. It then triggers the defence mechanism and notifies other instances of the system about dangerous event sequences.**

I. Introduction

Conventional protection against existing malware includes checking for known signatures of a virus in predefined places. While the conventional approach should undeniably be used to safeguard against known hazards, it does not provide protection against zero-day vulnerabilities, i.e. new attacks that have not been revealed yet, and do not have a released security patch yet. Their signature is not yet known to anti-virus programs, therefore the attack cannot be conclusively identified. The lack of self-learning and adaptation abilities of such systems also makes them reliant on external updates in order to keep signature knowledge base up to date.

One of the most important open research questions in cyber security is the ability of the system to intelligently detect new previously unseen threats, and react to them in such a way that minimizes the damage and potentially removes the threat altogether. Therefore, different artificial intelligence mechanisms, such as anomaly detection are widely used for this purpose. It is important, however, that the detection mechanisms should keep both types of mistakes (false negative and false positive) at minimum. If an existing attack is not detected (false negative), this allows the attacker to cause damage to the system unharmed. But if a legitimate behavior is detected as an attack (false positive), the system will try to stop the legitimate behavior, which in some cases can cause comparable amount of economical and operational damage to a non-detected genuine attack.

As shown in Figure 1, while signature-based methods usually have low false positive rate (they do not detect legitimate behavior as an attack), they inevitably have high false negative rate (they do not detect many actual attacks), due to not having attack signatures available for new attacks. Anomaly detection methods, on the other hand, have lower rate of false negatives, as they can detect even new attacks, but their false positives rate is largely increased, because any previously unseen behavior can be marked as an attack, even if it is legitimate.

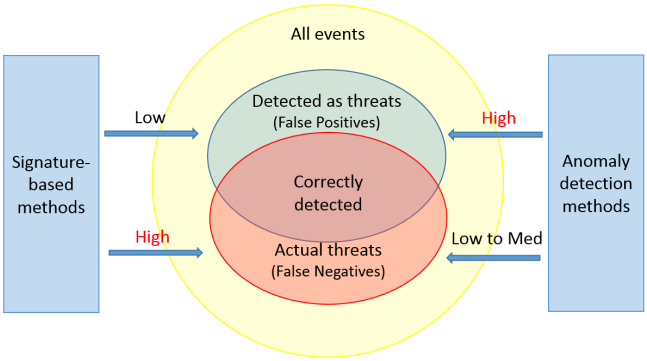

Figure 1. Signature-based vs. anomaly based methods of intrusion detection.

A system that features a full range of measures to identify and prevent attacks, and to recover from potential damage, is usually called *self-healing*. A good definition of a self-healing system is given in [1]: "*By self-healing systems, we understand this to be a resilient system able to carry out normal activities even when under attack; a system that can identify deviation from a 'normal' system and apply a corrective measure; a system that can identify intruders (parasites – external elements to the 'normal' network) and understand their impact, and a system that can reverse from new states to the 'normal state'.*"

This paper presents a proposed concept of an intrusion detection system based on anomaly detection and danger signals recognition. The system monitors events in the environment, constructs patterns of event sequences, and finds strange and anomalous patterns. If any dangerous symptoms are detected in the environment, the system matches them to the timeline of events and finds a pattern that may have caused the symptoms. The system then triggers the defence mechanism and notifies other instances of the system about dangerous event sequences.

II. Artificial Immune systems

Fuzzy intelligent mechanisms can be used to provide some level of protection against unseen zero-day attacks. Some investigation on this topic is done in the area of artificial immune systems (AIS) [2]. AIS looks at how biological mechanisms fight with unknown threats, e.g. an organism fights a new virus, and mimics it in a computer system. There are several biologically-inspired approaches that AIS state of the art research proposes. The most common one is a *negative selection* approach that is based on *anomaly* detection, another one is based on *danger theory*.

A. Negative Selection

Detection of anomalies in negative selection is usually called "non-self" detection [3], [4]. This is due to resemblance of the algorithm to how a biological organism recognises whether a cell is its own, or a foreign one. The main idea is to construct a set of "non-self" entities that do not pass a similarity test with any of pre-existing "self" entities. If a new entity is matched with any of these "non-self" entities, it is rejected as foreign. D'haeseleer [5] points out that negative selection method has properties of a successful immune system. The detection scheme is inherently distributable, every detector comparison can be done on a different host. Hosts can also have their own set of detectors. Negative selection requires no prior knowledge of intrusions, due to being in its core a general anomaly detection method. It is self-learning, as a set of detectors naturally evolves over time, when obsolete detectors die and new ones are obtained from the current event traffic. The set of detectors and the set of "self" entities are mutually protective: a change in one of them can be detected by looking at the other set.

B. Positive Selection

Similarly to negative selection, *positive selection* approach reduces the algorithm by one step, and instead of matching a new entity with a constructed "non-self" entity-set, it matches it with pre-existing "self" set, and rejects if no match is found. Negative selection is more widespread in the research of anomaly detection than positive selection. But some experiments suggest that these two approaches have similar performance. Dasgupta et al. [6] performed a series of experiments by using negative and positive selection techniques on the same data sets. Moreover, they examined difference in performance on an original data set and a reduced one, in order to understand the influence of information loss during detection phase. The study concludes that although both techniques produced similar results, further experiments need to be conducted in order to determine quantitative performance. In practice, the choice of positive versus negative selection approach comes down to which set (self or non-self) can be represented more concisely.

C. Negative and Positive Selection Limitations

The problem of non-self based intrusion detection mechanisms is that they bear a great false positive rate, as they trigger on any previously unseen information and behavior, whether it is legitimate or not. In some settings, such as fraud detection in banking systems, such approach leads to great results, because it can be reasonably expected that "self" detector cover all types of legitimate behavior and any anomaly ("non-self") is therefore a fraud. However, in network systems and many other types of systems, we can expect legitimate behavior to change over time, or to have some rare anomalous events that do not constitute an attack. Non-self negative selection for such intrusion detection produces great number of false positives whenever legitimate behavior changes. Therefore negative selection should be used on its own only in those systems, where no change of behavior over time is expected.

Another problem that was identified with negative selection is space scalability. Unlike many other anomaly detection methods that can extract useful features from a set of detectors, and only need to keep them, the negative selection approach needs to represent all possible detectors in explicit form. Kim and Bentley [7] performed experiments with a dataset of TCP connections. From every TCP connection they extracted a set of meaningful parameters that was used to encode detectors: connection identifiers, such as initiator's address and port and receiver's address and port; whether known vulnerabilities exist on either of connecting hosts; handshaking results; traffic intensity, etc. In total there were 33 different fields. Their experimentation showed that for such encoding of a detector, to achieve 80% detection rate they would need $6,4 * 10^8$ detectors.

D. Danger theory

In contrast to the negative selection approach, where every unknown entity raises an alarm, in *danger theory* (DT) approach immune responses are triggered by danger signals rather than just presence of non-self objects [8]. Any entities are allowed to exist in the body until harmful signals are received. If a harmful activity (e.g. cell death) is detected, the immune response is triggered, attacking either all foreign entities, or all entities locally, depending on the severity of the danger signal. Burgess [9] was among the first who proposed to use the biologically-inspired danger theory to detect and react on harmful activity in computer systems. However any harmful activity was considered, including random software and hardware errors. For network communication, signals such as packet-loss [10] are proposed to be defined as danger signals. Sarafijanovic and Boudec [10] provide a detailed description of a possible AIS system that is based on negative selection and danger signals. Their system is designed for a network routing setting. Every host has a separate detection system, which learns on its own, but has a possibility to receive events with learned information

from other neighboring nodes. Ou and Ou [11] describe the profiling of danger signals used to determine the threat profile of network packets and system calls. Danger signals may be composed of excessive CPU usage, memory load at the host, bandwidth saturation, high connection number of the host, etc. According to a threat profile, network packets receive an estimation of three parameters: attack severity, certainty, and the length of attacking time. They define an *antigen* as "an information vector extracted from network packet". Usually, antigens are binary strings, which include parameters, extracted from the IP packets, such as IP address, port number, protocol types, etc. Similarly, antigens can be constructed for system calls, or any other source of events. Aickelin et al. [12], [13] showed how Danger Theory establishes a link between Artificial Immune Systems and Intrusion Detection Systems. Several other works also investigated usage of danger theory in intrusion detection, proposing different variants of danger theory inspired AIS algorithms [14], [15], [16]. However, most works lack validation on proper real or realistic systems, and the approach that shows the best performance is yet to be found.

Danger signal signifies damage to the system, but not necessarily shows the origins of this damage and which entities are to blame. Therefore, the artificial immune system must decide, which entities may have caused this danger signal, and how to protect the system from further damage.

III. Danger Theory for Intrusion Detection

We propose to investigate an approach to create an intrusion detection self-healing system based on danger theory. We calculate an anomaly score for every event in the system. But being anomalous is not enough to trigger a response, as this may be a legitimate new behavior. There should be a known (or a highly probable) danger for the response to be triggered. As soon as the danger is known, the timeline of events is checked in order to find out either an anomalous sequence of events, a sequence that bears known danger risks, or a combination of the two. The corresponding defence response is then triggered, and the information about newly found dangerous sequence is sent to other machines in the network.

Every event has three computed values. The **type of event (ET)**, based on predefined types, or automated events clustering. This may also be in a form of similarity measure between two events. The **anomaly value (AV)**, how anomalous the event is, based on non-self computations. The **danger value (DV)**, which increases when any strange and potentially dangerous signals are associated with this event.

These three values are combined together in order to calculate the total **threat value (TV)**, the perceived potential of this event to cause damage or to be involved in a sequence of events that can cause damage to the system.

An instance of the system is running on every machine in the network and collects information about events. Depending on the purpose of the system, events can include network packets, system calls, CPU activity, etc. Instances are connected with each other, so it is possible to share information about detected dangers to other hosts.

There are three main system flows that are originated depending on the external trigger. When a **new event** is detected, it should be added to the timeline, which then should be checked for dangerous patterns, etc. When a **danger signal** is detected, the system must decide if any event pattern can be related to this danger signal and act accordingly. When a **warning** arrives from another host that carries information about a danger signal and related dangerous sequence, a host's own timeline should be checked to verify that it does not contain similar dangerous sequences.

A. New event analysis

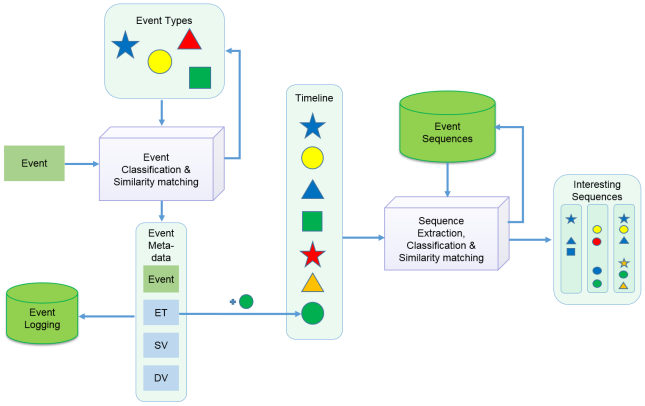

Figure 2. New event

The processing flow for registering a new event is described in Figure 2. The monitoring system is deployed to each machine in the network, and monitors all events. Possible events to monitor can include: system calls, network packets, port usage, emails, URL browsing, file downloads, memory usage, disk activity, etc. Algorithm for detecting events changes, depending on the network profile, and evolves with time. Event features are likely to change over time as well. For example, while originally botnets often used particular ports to establish connection with their peers [17], this served as an easy way to identify anomalous behavior. Therefore, increasingly botnets use common ports for communication, such as port 80 [18].

Every event is checked by the event classification system, to define its type and feature parameters. For a matching algorithm there is a possibility to match event on several levels to avoid unnecessary load. For example, we can define a "file change" event type, which matches any change in any file. On the highest level, two different changes in two different files are regarded as the same event. However,

during the investigation or more fine-grained matching, if there is a need to check, which files exactly were changed, the next level of matching checks the file name, extension, path, etc., and only considers events as similar if these parameters match. The third level may include the nature of the change itself, so the old version and the new version of both files should be compared to identify the exact changes.

Based on event analysis, its parameters are computed, such as its type, anomaly value and danger value. It should be noted that numerous anomaly detection algorithms provide different approaches to define the anomaly value of an event. Depending on the chosen approach, the calculated anomaly score can differ considerably, which has consequences on further reasoning when finding dangerous events. The event is then stored into a database for historical reference together with this enriched information.

A timeline of events is kept in real-time, and is updated with new events as soon as they are analysed. The timeline should only contain an event type or certain event characteristics, not the actual unique events, in order to be used in event sequence matching. Similar to event classification, a timeline of events is also analyzed, and related event sequences are extracted and classified. Classification system gets updated with new events, possibly changing classification when event profiles change. This need not happen in real time, and may happen as a batch job that runs periodically.

B. Danger signal processing

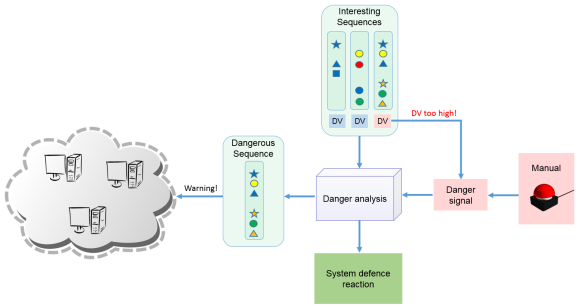

Figure 3. Danger signal

Reaction on a danger signal is described in Figure 3.

A danger signal is a sign to the system that something adverse has happened, which either caused damage to the system or has a high potential to cause it.

We investigate two possibilities for the system to receive a danger signal. The first is manual, i.e. a system's operator or manager manually notifies the system of observed danger symptoms and triggers defence protocols. The second is automatic, which includes detecting a certain event or a combination of events that is predefined to be intrinsically dangerous. For example, such signals can be: network packet loss, anomalous packets, usage of strange ports, anomalous memory or CPU usage, inappropriate disk activity, inappropriate user actions. These symptoms can be of different severity, and may be regarded possibly including known relations of such symptoms to previous timeline of events. The system immediately triggers the danger processing mode when it detects such events. It is possible that a sequence of events, each being an admissible event but with a certain danger value, together constitute a greater danger, which exceeds the required threat threshold. When a danger signal is obtained, the system knows that there is an attack, and tries to find its origins and prevent it from spreading.

As an example of what the system's defence reaction can be, when a dangerous trace is found: the machine can be prevented to communicate with the network, or to execute certain system calls which may result in further damage (i.e. disk overwriting, or sending commands to PLCs, as those that resulted in centrifuges failure in Stuxnet virus).

For partial sequence matching on other machines, the minimal reaction (to avoid overreacting) is to forbid executing only those actions, that will further match the dangerous sequence, but allow execution of all other commands, while creating an alert of possible danger.

Every dangerous sequence is also added to the database of identified dangerous sequences for future reference.

C. Warning signal processing

Receiving a warning signal is described in Figure 4.

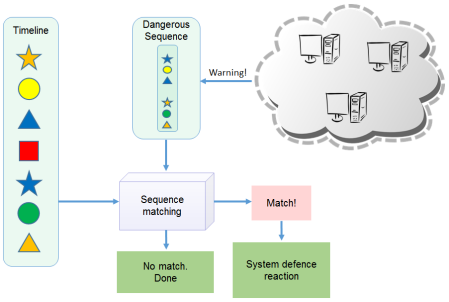

Figure 4. Warning signal

A warning signal comes from a network of trusted machines. If some machine has detected an attack and identified (possibly) responsible sequences of events, these sequences are sent to all other machines in the network. They check their own timeline to identify, if they have experienced similar events or a certain prefix of the sequence of events, because an attack can be in earlier stages. Also, the variability of event matching should be greater on a different machine, due to the fact that the same infection may come from a different source or have different parameters. For partial sequence matching of sequences coming from other machines, the minimal reaction (to avoid overreacting) can be to forbid executing only those actions that will further match the dangerous sequence, but allow execution

of all other commands, while notifying administrators of a possible danger. It is also possible that when the timeline is updated, it gets a match with a sequence from a database of dangerous sequences. If there is a match of any dangerous sequence with the timeline, a defence reaction is activated. If there is no match, the system continues to operate normally.

IV. Scenario

Worms are self replicating programs. One such malware, known as *Conficker*, generates a very large amount of network traffic that overwhelms communicating devices. An example of the chaos created by Conficker occurred in 2011 at a steel plant: *On February 6, 2011, the ALSPA system stopped. An investigation revealed that there was a Conficker virus infection in all machines of the ALSPA system. The worm spread throughout the power plant automation network (and probably in other automation networks [...]). The virus flooded the network with unwanted packets and caused an instability in the communications between PLCs and supervisory stations and freezing most of the supervisory systems. The automation team cleaned the infected machines, but the virus returned*[19].

Let us consider a scenario that illustrates how self-healing network intrusion detection system can automatically stop the spread of worm malware. With the self-healing functionality running as part of a computer's Operating System (OS) scheduler, program activity is used as a source of pattern data for an automatic danger signal generation module to analyse. Thus, the danger signal is derived from monitoring system performance metrics such as outgoing network traffic and thread activity etc. The computer is connected to other computers on a network and each computer's OS also runs the self-healing functionality as part of its scheduler. The network of computers function normally and the danger signal generation modules within the OSs build patterns of typical activity. New software is installed, updates and patches occur, and no danger signals are generated.

One computer becomes infected with the Conficker-like worm. It adds e.g., *Proc-X* to its services and the malware starts its mischief. As a side effect of the malware execution, outgoing network traffic increases and eventually after time t_{tr}, the danger signal is triggered. Over time t_{co}, the automatic danger signal generation module on the infected computer correlates the pattern of program activity with the addition of Proc-X in its services list, and broadcasts a compact message into the network warning its peers of the problem. Upon receiving this message, each computer takes time t_{im} to immunise itself by halting processor cycle time given to Proc-X execution. As the worm takes time t_{inf} to infect a computer, arguably it will take time $k.t_{inf}$ to infect the whole network, for some value k. Therefore, if $t_{tr} + t_{co} + t_{im} < k.t_{inf}$ then propagation of the worm will be halted as computers on the network take action and immunise themselves. This is illustrated in Figure 5.

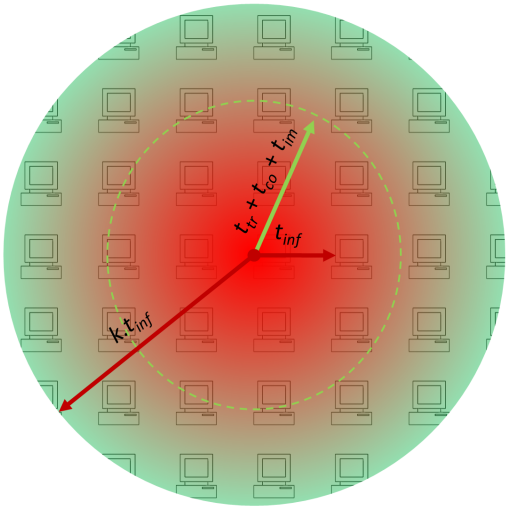

Figure 5. The self-healing network intrusion detection system preventing spread of worm malware over time. The dotted line indicates the boundary at which the worm is stopped as all computers are immunised against it.

V. Conclusions

In this paper, we presented a concept of self-healing network intrusion detection system based on danger theory. Unlike traditional signature-based methods, this approach can defend from previously unknown attacks. Also, unlike pure anomaly detection methods that react on any new behavior in the system, this approach allows to distinguish a new legitimate behavior from a new dangerous behavior, and only react on dangerous ones. This allows to drastically reduce the number of false positive alarms in highly-dynamic systems, where new behaviors emerge constantly. Dangerous events are shared over the network, allowing to quickly propagate defence reaction to other servers, and stop the attack on other machines at its earlier stages.

References

[1] A. Alhomoud, I. Awan, and J. P. Disso, "Towards an enterprise self-healing system against botnets attacks," in *Computing, Networking and Communications (ICNC), 2013 International Conference on*. IEEE, 2013, pp. 1112–1117.

[2] J. Kim, P. J. Bentley, U. Aickelin, J. Greensmith, G. Tedesco, and J. Twycross, "Immune system approaches to intrusion detection–a review," *Natural computing*, vol. 6, no. 4, pp. 413–466, 2007.

[3] S. Forrest, A. S. Perelson, L. Allen, and R. Cherukuri, "Self-nonself discrimination in a computer," in *2012 IEEE Symposium on Security and Privacy*. IEEE Computer Society, 1994, pp. 202–202.

[4] A. Somayaji, S. Hofmeyr, and S. Forrest, "Principles of a computer immune system," in *Proceedings of the 1997 workshop on New security paradigms*. ACM, 1998, pp. 75–82.

[5] P. D'haeseleer, "An immunological approach to change detection: Theoretical results," in *Computer Security Foundations Workshop, 1996. Proceedings., 9th IEEE*. IEEE, 1996, pp. 18–26.

[6] D. Dasgupta and F. Nino, "A comparison of negative and positive selection algorithms in novel pattern detection," in *Systems, Man, and Cybernetics, 2000 IEEE International Conference on*, vol. 1. IEEE, 2000, pp. 125–130.

[7] J. Kim and P. J. Bentley, "An evaluation of negative selection in an artificial immune system for network intrusion detection," in *Proceedings of GECCO*, 2001, pp. 1330–1337.

[8] P. Matzinger, "Tolerance, danger, and the extended family," *Annual review of immunology*, vol. 12, no. 1, pp. 991–1045, 1994.

[9] M. Burgess *et al.*, "Computer immunology." in *Systems Administration Conference (LISA)*, vol. 98, 1998, pp. 283–298.

[10] S. Sarafijanović and J.-Y. Le Boudec, "An artificial immune system for misbehavior detection in mobile ad-hoc networks with virtual thymus, clustering, danger signal, and memory detectors," in *Artificial Immune Systems*. Springer, 2004, pp. 342–356.

[11] C.-M. Ou and C.-R. Ou, "Immunity-inspired host-based intrusion detection systems," in *Genetic and Evolutionary Computing (ICGEC), 2011 Fifth International Conference on*. IEEE, 2011, pp. 283–286.

[12] U. Aickelin and S. Cayzer, "The danger theory and its application to artificial immune systems," in *Second Internation Conference on Artificial Immune Systems (ICARIS-02)*, 2002, pp. 141–148.

[13] U. Aickelin, P. Bentley, S. Cayzer, J. Kim, and J. McLeod, "Danger theory: The link between ais and ids?" in *Artificial Immune Systems*. Springer, 2003, pp. 147–155.

[14] M. Swimmer, "Using the danger model of immune systems for distributed defense in modern data networks," *Computer Networks*, vol. 51, no. 5, pp. 1315–1333, 2007.

[15] H. Fu, X. Yuan, and N. Wang, "Multi-agents artificial immune system (maais) inspired by danger theory for anomaly detection," in *Computational Intelligence and Security Workshops, 2007. CISW 2007. International Conference on*. IEEE, 2007, pp. 570–573.

[16] J. Kim, J. Greensmith, J. Twycross, and U. Aickelin, "Malicious code execution detection and response immune system inspired by the danger theory," 2010, arXiv.

[17] J. F. P. Disso, K. Jones, P. Williams, and A. Steer, "A distributed attack detection and mitigation framework," in *Internet Multimedia Systems Architecture and Application (IMSAA), 2011 IEEE 5th International Conference on*. IEEE, 2011, pp. 1–6.

[18] M. Cremonini and M. Riccardi, "The dorothy project: An open botnet analysis framework for automatic tracking and activity visualization," in *Computer Network Defense (EC2ND), 2009 European Conference on*. IEEE, 2009, pp. 52–54.

[19] RISI. Steel plant infected with conficker. Repository of Industrial Security Incidents (RISI). [Online]. Available: http://www.risidata.com/Database/Search_Results/search&keywords=conficker/